USER MANUAL

Version 1.0

Contents

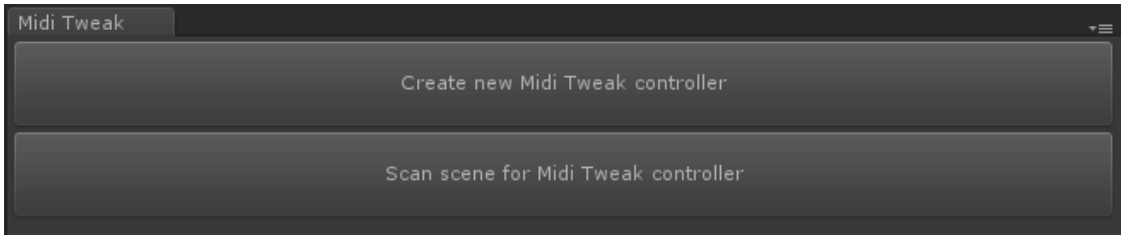
Midi Tweak Setup

The core functionality of Midi Tweak is to make it easier to fine tune specific variables within a game. To start tweaking variables within the game, they will have to be added to a list of parameters.

All scene related parameters are stored within a “Midi Tweak controller” game object that is added to the scene. This game object can easily be created through the Midi Tweak editor window.

1. Open Midi Tweak by going to Window > Midi Tweak within the Unity editor.

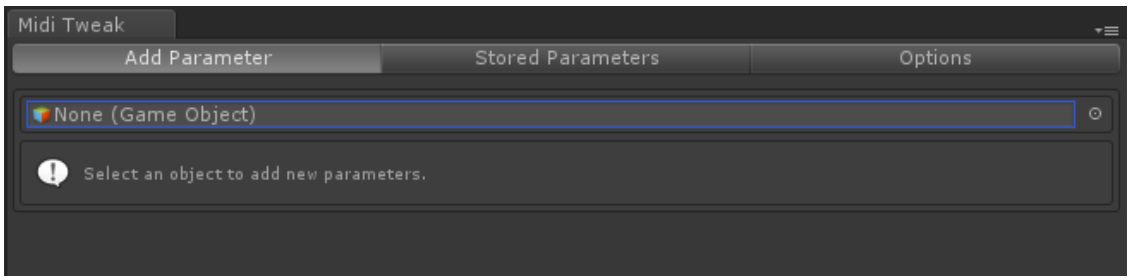
2. Press Create new Midi Tweak controller.



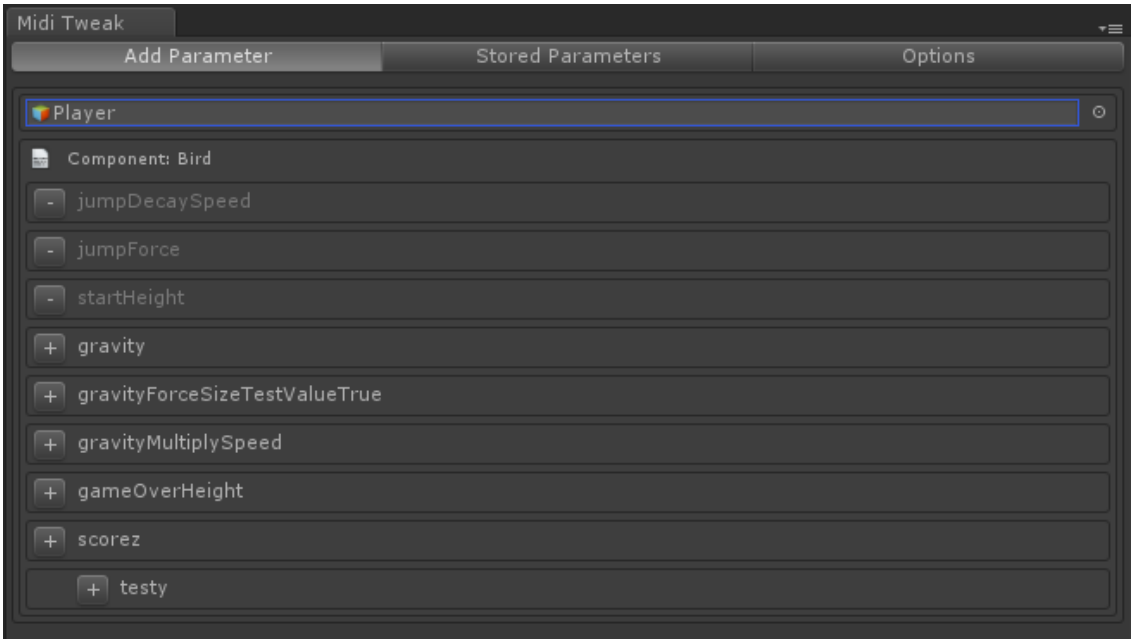
Scroll down to get an overview of all tabs.

Add Parameter Tab

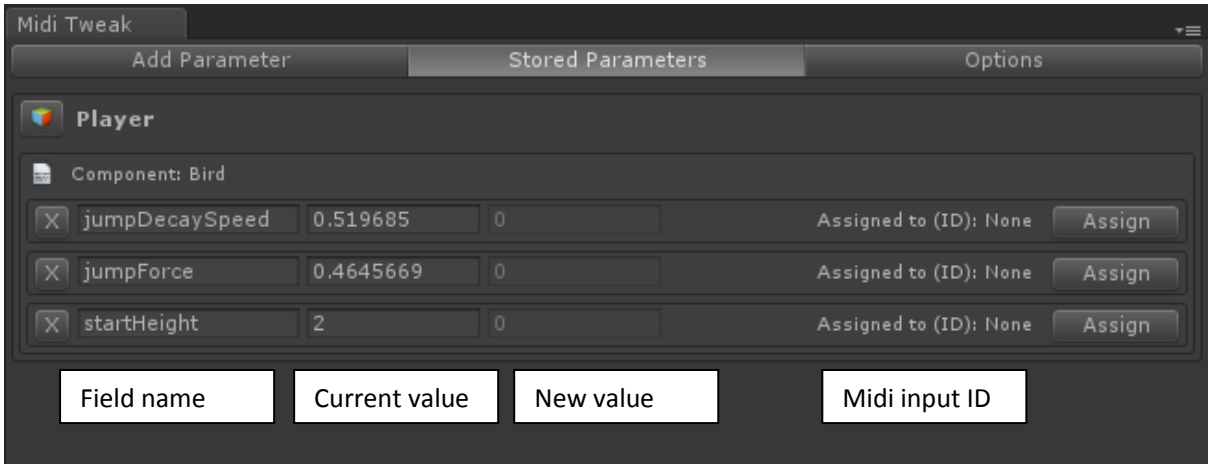
Add Parameters – This tab is used to add parameters to the stored parameters list



Select an object to see what parameters you can add to the list
(This is currently limited to fields of of type int or float)



Stored Parameters Tab



Here you can see all stored parameters. With the ability to remove them from the list, modify the values and assign a specific midi index to a variable.

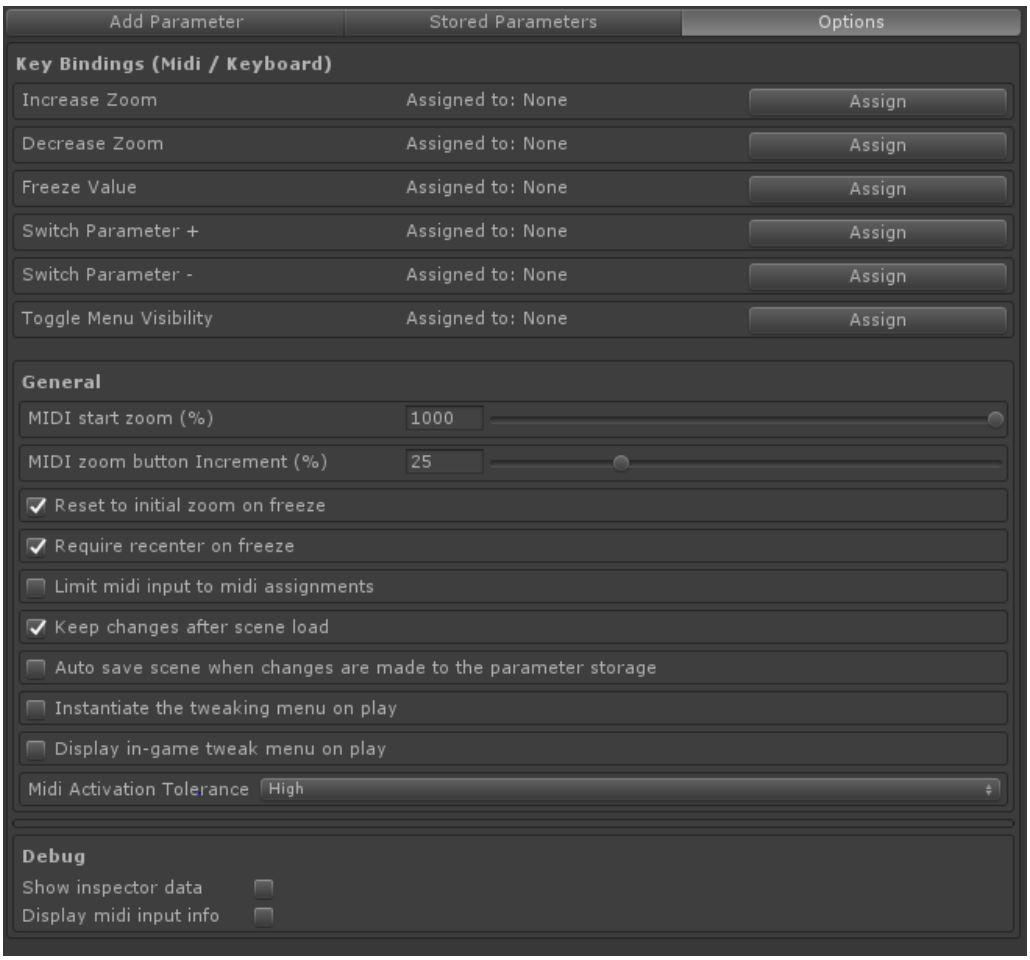
- **Field name**
  This is the name of the field of the variable.

- **Current value**
  This is the current value. You can modify it within the editor.

- **New value.**
  This is filled in after a play session. It will display your tweaked value.
  You will be able to apply it to the object.

- **Midi Input ID**
  This is the index based on the knob you used when assigning.

Options Tab



Bindings

- **Increase zoom**
  This increases the zoom by the MIDI zoom button increment (%) displayed below.

- **Decrease zoom**
  This decreases the zoom by the MIDI zoom button increment (%) displayed below.

- **Freeze value**
  Sets the current value as the base

- **Switch parameter forward**
  Switch to the next parameter

- **Switch parameter backward**
  Switch to previous parameter

- **Toggle menu visibility**
  Toggle the visibility of the midi tweak indicator

General options

- **Midi start zoom**
  Starting zoom range

- **Midi zoom button increment**
  The amount the zoom should decrease or increase,
  when pressing the increase zoom or decrease zoom hotkey.

- **Reset to initial zoom on freeze**
  This will reset the zoom to the start zoom once the value is frozen.

- **Require center on freeze**
  This requires you to center the input(knob) after freezing. Before being able to modify values again.

- **Limit midi input to midi assignments**
  Ensures that you cannot use any other midi knobs except than the ones you specified within the stored parameters tab.

- **Keep changes after scene load**
  Will not reset any changes after loading a new scene. This is beneficial for prototypes or games that require extensive scene changing.

- **Auto save scene when changes are made to the parameter storage**
  Parameters are stored within a game object in the scene. Enabling this will automatically save the scene for you once you add/remove or set any parameters.

- **Instantiate the tweaking menu on play**
  This will spawn the tweaking menu within your scene.

- **Display in-game tweak menu on play**
  Show the tweak menu directly when pressing play. If you untoggle this, you can still display it through the “Toggle menu visibility” hotkey. Ensure that instantiate the tweaking menu on play is toggled on.

- **Midi activation tolerance**
  Set this to make it harder to switch between midi knobs. This is to prevent accidental movement on knobs.

Tweak tab

Parameters
You can see all added parameters here. With the included ability to select different parameters.

Bindings
Displays all the bindings information.

Tweak Tools
Everything you need to tweak your variables is within this section. With the option to:
* Modify values with slider
* Freeze the value with button
* Set the zoom
* Set a new mid-value

API Reference

- MidiTweak. Instance.SetValueByPercentage(percentage, opt index)
  Set the current tweak value by percentage (0f to 1.0f)

- MidiTweak. Instance.SetValue(value, opt index)
  Set the current tweak value by value

- MidiTweak. Instance.SwitchParameter(direction)
  Switch the active parameter to the specified direction (+ = forward, - = backwards)

- MidiTweak. Instance.SetActiveParameter(parameterIndex, midiKeyIndex = 0, updateZoomRange = true, setFrozen = true)

- MidiTweak. Instance. GetActiveBookmark()
  Get current bookmark data. Which contains the following data:
  knobID – midi input index that is attached to the parameter
  component – component that is being modified with this parameter
  field – field that is being modified with this parameter

- MidiTweak. Instance. GetActiveBookmarkIndex()
  Gets the index of the currently selected bookmark

- MidiTweak. Instance.SetFrozen(state)
  Sets the current value frozen/unfrozen based on “state”
  A frozen state means that the current value becomes the mid value.
  And that you will need to re-center the knob (based on the configuration)

- MidiTweak. Instance.SetZoomRange(range)
  Set the zoom range. Passing 100 will set the range to -100% and 100%.

- MidiTweak. Instance.IncreaseZoomRange(amount)
  Increases the zoom range by an amount.

- MidiTweak. Instance.FreezeCurrentValue()
  Freezes the current value. Which means that the current value will become the new mid-point.

- MidiTweak. Instance.ToggleDisplayUserInterface()
  Toggles the visibility of the tweak user interface.

About

This plugin has been created and developed by Lowscope.
www.low-scope.com.

For any support or questions please email:
info@low-scope.com

For more information on usage rights please refer to:
https://unity3d.com/legal/as_terms

We hope you enjoy the product. And if there are any issues or requests feel free to send us an email. We are happy to help.

Midi Jack – External Library

Midi Tweak depends greatly on Midi Jack for processing midi inputs.
If your intention is purely to receive specific midi keys, then it is recommended to do this through the Midi Jack plugin.

Source of the following information: https://github.com/keijiro/MidiJack

API Reference – Midi Jack

The basic functions of MIDI Jack are provided in the MidiMaster class.

The channel arguments in the following functions can be omitted. In that case, it returns the value in the All-Channel slot, which stores mixed status of all active channels.

- MidiMaster.GetKey (channel, noteNumber)
  Returns the velocity value while the key is pressed, or zero while the key is released. The value ranges from 0.0 (note-off) to 1.0 (maximum velocity).

- MidiMaster.GetKeyDown (channel, noteNumber)
  Returns true during the frame the user starts pressing down the key.

- MidiMaster.GetKeyUp (channel, noteNumber)
  Returns true during the frame the user releases the key.

- MidiMaster.GetKnob (channel, knobNumber, defaultValue)
  Returns the controller value (CC). The value ranges from 0.0 to 1.0.

- MidiMaster.GetKnobNumbers (channel)
  Returns the list of active controllers.

There are also delegates for the each type of MIDI event.

- MidiMaster.noteOnDelegate (channel, noteNumber, velocity)
- MidiMaster.noteOffDelegate (channel, noteNumber)
- MidiMaster.knobDelegate (channel, knobNumber, konbValue)

Current Limitations – Midi Jack

- Currently MIDI Jack only supports Windows and OS X. No iOS support yet.
- Only supports note and CC messages. No support for program changes nor SysEx.
- The MIDI Jack plugin always tries to capture all available MIDI devices. On Windows this behavior may conflict with other MIDI applications.

License - Midi Jack